

Reconfiguration over tree decompositions

Amer E. Mouawad^{1*}, Naomi Nishimura^{1*}, Venkatesh Raman², and
Marcin Wrochna³

¹ David R. Cheriton School of Computer Science
University of Waterloo, Ontario, Canada.

{aabdomou, nishi}@uwaterloo.ca

² Institute of Mathematical Sciences
Chennai, India.

vraman@imsc.res.in

³ Institute of Computer Science
Uniwersytet Warszawski, Warsaw, Poland.

mw290715@students.mimuw.edu.pl

Abstract. A vertex-subset graph problem Q defines which subsets of the vertices of an input graph are feasible solutions. The reconfiguration version of a vertex-subset problem Q asks whether it is possible to transform one feasible solution for Q into another in at most ℓ steps, where each step is a vertex addition or deletion, and each intermediate set is also a feasible solution for Q of size bounded by k . Motivated by recent results establishing $W[1]$ -hardness of the reconfiguration versions of most vertex-subset problems parameterized by ℓ , we investigate the complexity of such problems restricted to graphs of bounded treewidth. We show that the reconfiguration versions of most vertex-subset problems remain PSPACE-complete on graphs of treewidth at most t but are fixed-parameter tractable parameterized by $\ell + t$ for all vertex-subset problems definable in monadic second-order logic (MSOL). To prove the latter result, we introduce a technique which allows us to circumvent cardinality constraints and define reconfiguration problems in MSOL.

1 Introduction

Reconfiguration problems allow the study of structural and algorithmic questions related to the solution space of computational problems, represented as a *reconfiguration graph* where feasible solutions are represented by nodes and adjacency by edges [7, 17, 19]; a path is equivalent to the step-by-step transformation of one solution into another as a *reconfiguration sequence* of *reconfiguration steps*.

Reconfiguration problems have so far been studied mainly under classical complexity assumptions, with most work devoted to deciding whether it is possible to find a path between two solutions. For several problems, this question has been shown to be PSPACE-complete [5, 19, 20], using reductions that construct

* Research supported by the Natural Science and Engineering Research Council of Canada.

examples where the length ℓ of reconfiguration sequences can be exponential in the size of the input graph. It is therefore natural to ask whether we can achieve tractability if we allow the running time to depend on ℓ or on other properties of the problem, such as a bound k on the size of feasible solutions. These results motivated Mouawad et al. [23] to study reconfiguration under the *parameterized complexity* framework [13], showing the $W[1]$ -hardness of VERTEX COVER RECONFIGURATION (VC-R), FEEDBACK VERTEX SET RECONFIGURATION (FVS-R), and ODD CYCLE TRANSVERSAL RECONFIGURATION (OCT-R) parameterized by ℓ , and of INDEPENDENT SET RECONFIGURATION (IS-R), INDUCED FOREST RECONFIGURATION (IF-R), and INDUCED BIPARTITE SUBGRAPH RECONFIGURATION (IBS-R) parameterized by $k + \ell$ [23].

Here we focus on reconfiguration problems restricted to \mathcal{C}_t , the class of graphs of treewidth at most t . In Section 3, we show that a large number of reconfiguration problems, including the six aforementioned problems, remain PSPACE-complete on \mathcal{C}_t , answering a question left open by Bonsma [6]. The result is in fact stronger in that it applies to graphs of bounded bandwidth and even to the question of finding a reconfiguration sequence of *any* length.

In Section 4, using an adaptation of Courcelle’s cornerstone result [9], we present a meta-theorem proving that the reconfiguration versions of all vertex-subset problems definable in monadic second-order logic become tractable on \mathcal{C}_t when parameterized by $\ell + t$. Since the running times implied by our meta-theorem are far from practical, we consider the reconfiguration versions of problems defined in terms of hereditary graph properties in Section 5. In particular, we first introduce signatures to succinctly represent reconfiguration sequences and define “generic” procedures on signatures which can be used to exploit the structure of nice tree decompositions. We use these procedures in Section 5.2 to design algorithms solving VC-R and IS-R in $\mathcal{O}^*(4^\ell(t+3)^\ell)$ time (the \mathcal{O}^* notation suppresses factors polynomial in n , ℓ , and t). In Section 5.4, we extend the algorithms to solve OCT-R and IBS-R in $\mathcal{O}^*(2^{\ell t}4^\ell(t+3)^\ell)$ time, as well as FVS-R and IF-R in $\mathcal{O}^*(t^{\ell t}4^\ell(t+3)^\ell)$ time. We further demonstrate in Section 5.3 that VC-R and IS-R parameterized by ℓ can be solved in $\mathcal{O}^*(4^\ell(3\ell+2)^\ell)$ time on planar graphs by an adaptation of Baker’s shifting technique [1].

2 Preliminaries

For general graph theoretic definitions, we refer the reader to the book of Diestel [12]. We assume that each input graph G is a simple undirected graph with vertex set $V(G)$ and edge set $E(G)$, where $|V(G)| = n$ and $|E(G)| = m$. The *open neighborhood* of a vertex v is denoted by $N_G(v) = \{u \mid uv \in E(G)\}$ and the *closed neighborhood* by $N_G[v] = N_G(v) \cup \{v\}$. For a set of vertices $S \subseteq V(G)$, we define $N_G(S) = \{v \notin S \mid uv \in E(G), u \in S\}$ and $N_G[S] = N_G(S) \cup S$. We drop the subscript G when clear from context. The subgraph of G induced by S is denoted by $G[S]$, where $G[S]$ has vertex set S and edge set $\{uv \in E(G) \mid u, v \in S\}$. Given two sets $S_1, S_2 \subseteq V(G)$, we let $S_1 \Delta S_2 = \{S_1 \setminus S_2\} \cup \{S_2 \setminus S_1\}$ denote the symmetric difference of S_1 and S_2 .

We say a graph problem Q is a *vertex-subset* problem whenever feasible solutions for Q on input G correspond to subsets of $V(G)$. Q is a *vertex-subset minimization (maximization)* problem whenever feasible solutions for Q correspond to subsets of $V(G)$ of size at most (at least) k , for some integer k . The *reconfiguration graph* of a vertex-subset minimization (maximization) problem Q , $R_{\min}(G, k)$ ($R_{\max}(G, k)$), has a node for each $S \subseteq V(G)$ such that $|S| \leq k$ ($|S| \geq k$) and S is a feasible solution for Q . We say k is the *maximum (minimum) allowed capacity* for $R_{\min}(G, k)$ ($R_{\max}(G, k)$). Nodes in a reconfiguration graph are adjacent if they differ by the addition or deletion of a single vertex.

Definition 1. For any vertex-subset problem Q , graph G , positive integers k and ℓ , $S_s \subseteq V(G)$, and $S_t \subseteq V(G)$, we define four decision problems:

- $Q\text{-MIN}(G, k)$: Is there $S \subseteq V(G)$ such that $|S| \leq k$ and S is a feasible solution for Q ?
- $Q\text{-MAX}(G, k)$: Is there $S \subseteq V(G)$ such that $|S| \geq k$ and S is a feasible solution for Q ?
- $Q\text{-MIN-R}(G, S_s, S_t, k, \ell)$: For $S_s, S_t \in V(R_{\min}(G, k))$, is there a path of length at most ℓ between the nodes for S_s and S_t in $R_{\min}(G, k)$?
- $Q\text{-MAX-R}(G, S_s, S_t, k, \ell)$: For $S_s, S_t \in V(R_{\max}(G, k))$, is there a path of length at most ℓ between the nodes for S_s and S_t in $R_{\max}(G, k)$?

For ease of description, we present our positive results for paths of length exactly ℓ , as all our algorithmic techniques can be generalized to shorter paths. Throughout, we implicitly consider reconfiguration problems as parameterized problems with ℓ as the parameter. The reader is referred to the books of Downey and Fellows [13], Flum and Grohe [16], and Niedermeier [24] for more on parameterized complexity.

In Section 5, we consider problems that can be defined using graph properties, where a *graph property* Π is a collection of graphs closed under isomorphism, and is *non-trivial* if it is non-empty and does not contain all graphs. A graph property is *polynomially decidable* if for any graph G , it can be decided in polynomial time whether G is in Π . The property Π is *hereditary* if for any $G \in \Pi$, any induced subgraph of G is also in Π . For a graph property Π , $R_{\max}(G, k)$ has a node for each $S \subseteq V(G)$ such that $|S| \geq k$ and $G[S]$ has property Π , and $R_{\min}(G, k)$ has a node for each $S \subseteq V(G)$ such that $|S| \leq k$ and $G[V(G) \setminus S]$ has property Π . We use $\Pi\text{-MIN-R}$ and $\Pi\text{-MAX-R}$ instead of $Q\text{-MIN-R}$ and $Q\text{-MAX-R}$, respectively, to denote *reconfiguration problems for Π* ; examples include VC-R, FVS-R, and OCT-R for the former and IS-R, IF-R, and IBS-R for the latter, for Π defined as the collection of all edgeless graphs, forests, and bipartite graphs, respectively.

Proofs of propositions, lemmas, and theorems marked with a star can be found in the appendix.

Proposition 2. Given Π and a collection of graphs \mathcal{C} , if $\Pi\text{-MIN-R}$ parameterized by ℓ is fixed-parameter tractable on \mathcal{C} then so is $\Pi\text{-MAX-R}$.

Proof. Given an instance (G, S_s, S_t, k, ℓ) of Π -MAX-R, where $G \in \mathcal{C}$, we solve the Π -MIN-R instance $(G, V(G) \setminus S_s, V(G) \setminus S_t, n-k, \ell)$. Note that the parameter ℓ remains unchanged.

It is not hard to see that there exists a path between the nodes corresponding to S_s and S_t in $R_{\text{MAX}}(G, k)$ if and only if there exists a path of the same length between the nodes corresponding to $V(G) \setminus S_s$ and $V(G) \setminus S_t$ in $R_{\text{MIN}}(G, n-k)$. \square

We obtain our results by solving Π -MIN-R, which by Proposition 2 implies results for Π -MAX-R. We always assume Π to be non-trivial, polynomially decidable, and hereditary.

Our algorithms rely on dynamic programming over graphs of bounded treewidth. A *tree decomposition* of a graph G is a pair $\mathcal{T} = (T, \chi)$, where T is a tree and χ is a mapping that assigns to each node $i \in V(T)$ a vertex subset X_i (called a *bag*) such that: (1) $\bigcup_{i \in V(T)} X_i = V(G)$, (2) for every edge $uv \in E(G)$, there exists a node $i \in V(T)$ such that the bag $\chi(i) = X_i$ contains both u and v , and (3) for every $v \in V(G)$, the set $\{i \in V(T) \mid v \in X_i\}$ forms a connected subgraph (subtree) of T . The *width* of any tree decomposition \mathcal{T} is equal to $\max_{i \in V(T)} |X_i| - 1$. The *treewidth* of a graph G , $tw(G)$, is the minimum width of a tree decomposition of G .

For any graph of treewidth t , we can compute a tree decomposition of width t and transform it into a nice tree decomposition of the same width in linear time [22], where a rooted tree decomposition $\mathcal{T} = (T, \chi)$ with root $root$ of a graph G is a *nice tree decomposition* if each of its nodes is either (1) a leaf node (a node i with $|X_i| = 1$ and no children), (2) an introduce node (a node i with exactly one child j such that $X_i = X_j \cup \{v\}$ for some vertex $v \notin X_j$; v is said to be *introduced* in i), (3) a forget node (a node i with exactly one child j such that $X_i = X_j \setminus \{v\}$ for some vertex $v \in X_j$; v is said to be *forgotten* in i), or (4) a join node (a node i with two children p and q such that $X_i = X_p = X_q$). For node $i \in V(T)$, we use T_i to denote the subtree of T rooted at i and V_i to denote the set of vertices of G contained in the bags of T_i . Thus $G[V_{root}] = G$.

3 PSPACE-completeness

We define a simple intermediary problem that highlights the essential elements of a PSPACE-hard reconfiguration problem. Given a pair $H = (\Sigma, E)$, where Σ is an alphabet and $E \subseteq \Sigma^2$ a binary relation between symbols, we say that a word over Σ is an *H-word* if every two consecutive symbols are in the relation. If one looks at H as a digraph (possibly with loops), a word is an *H-word* if and only if it is a walk in H . The *H-WORD RECONFIGURATION* problem asks whether two given *H-words* of equal length can be transformed into one another (in any number of steps) by changing one symbol at a time so that all intermediary steps are also *H-words*.

A *Thue system* is a pair (Σ, R) , where Σ is a finite alphabet and $R \subseteq \Sigma^* \times \Sigma^*$ is a set of rules. A rule can be applied to a word by replacing one subword by

the other, that is, for two words $s, t \in \Sigma^*$, we write $s \leftrightarrow_R t$ if there is a rule $\{\alpha, \beta\} \in R$ and words $u, v \in \Sigma^*$ such that $s = u\alpha v$ and $t = u\beta v$. The reflexive transitive closure of this relation defines an equivalence relation \leftrightarrow_R^* , where words s, t are equivalent if and only if one can be reached from the other by repeated application of rules. The *word problem* of R is the problem of deciding, given two words $s, t \in \Sigma^*$, whether $s \leftrightarrow_R^* t$. A Thue system is called *c-balanced* if for each $\{\alpha, \beta\} \in R$ we have $|\alpha| = |\beta| = c$. The following fact is a folklore variant [2] of the classic proof of undecidability for general Thue systems [25].

Lemma 3 (*). *There exists a 2-balanced Thue system whose word problem is PSPACE-complete.*

A simple but technical reduction from Lemma 3 allows us to show the PSPACE-completeness of H -WORD RECONFIGURATION. The simplicity of the problem statement allows for easy reductions to various reconfiguration problems, as exemplified in Theorem 5. Similar reductions apply to the reconfiguration versions of, e.g., k -COLORING [8] and SHORTEST PATH [21] – a comprehensive discussion is available in an online manuscript by the fourth author [27].

Lemma 4 (*). *There exists a digraph H for which H -WORD RECONFIGURATION is PSPACE-complete.*

Theorem 5. *There exists an integer b such that VC-R, FVS-R, OCT-R, IS-R, IF-R, and IBS-R are PSPACE-complete even when restricted to graphs of treewidth at most b .*

Proof. Let $H = (\Sigma, R)$ be the digraph obtained from Lemma 4. We show a reduction from H -WORD RECONFIGURATION to VC-R.

For an integer n , we define G_n as follows. The vertex set contains vertices v_i^a for all $i \in \{1, \dots, n\}$ and $a \in \Sigma$. Let $V_i = \{v_i^a \mid a \in \Sigma\}$ for $i \in \{1, \dots, n\}$. The edge set of G_n contains an edge between every two vertices of V_i for $i \in \{1, \dots, n\}$ and an edge $v_i^a v_{i+1}^b$ for all $(a, b) \notin R$ and $i \in \{1, \dots, n-1\}$. The sets $V_i \cup V_{i+1}$ give a tree decomposition of width $b = 2|\Sigma|$.

Let $k = n \cdot (|\Sigma| - 1)$ and consider a vertex cover S of G_n of size k . For all i , since $G_n[V_i]$ is a clique, S contains all vertices of V_i except at most one. Since $|S| = \sum_i (|V_i| - 1)$, S contains all vertices except exactly one from each set V_i , say $v_i^{s_i}$ for some $s_i \in \Sigma$. Now $s_1 \dots s_n$ is an H -word ($s_i s_{i+1} \in R$, as otherwise $v_i^{s_i} v_{i+1}^{s_{i+1}}$ would be an uncovered edge) and any H -word can be obtained in a similar way, giving a bijection between vertex covers of G_n of size k and H -words of length n .

Consider an instance $s, t \in \Sigma^*$ of H -WORD RECONFIGURATION. We construct the instance $(G_n, S_s, S_t, k+1, \ell)$ of VC-R, where $n = |s| = |t|$, $\ell = 2^{n|\Sigma|}$ (that is, we ask for a reconfiguration sequence of any length) and S_s and S_t are the vertex covers of size k that correspond to s and t , respectively. Any reconfiguration sequence between such vertex covers starts by adding a vertex (since G_n has no vertex cover of size $k-1$) and then removing another (since vertex covers larger than $k+1$ are not allowed), which corresponds to changing one symbol of

an H -word. This gives a one-to-one correspondence between reconfiguration sequences of H -words and reconfiguration sequences (of exactly twice the length) between vertex covers of size k . The instances are thus equivalent.

This proof can be adapted to FVS-R and OCT-R by replacing edges with cycles, e.g. triangles [23]. For IS-R, IF-R, and IBS-R, we simply need to consider set complements of solutions for VC-R, FVS-R, and OCT-R, respectively. \square

4 A meta-theorem

In contrast to Theorem 5, in this section we show that a host of reconfiguration problems definable in monadic second-order logic (MSOL) become fixed-parameter tractable when parameterized by $\ell + t$. First, we briefly review the syntax and semantics of MSOL over graphs. The reader is referred to the excellent survey by Martin Grohe [18] for more details.

We have an infinite set of *individual variables*, denoted by lowercase letters x, y , and z , and an infinite set of *set variables*, denoted by uppercase letters X, Y , and Z . A *monadic second-order formula (MSOL-formula)* ϕ over a graph G is constructed from *atomic formulas* $\mathcal{E}(x, y)$, $x \in X$, and $x = y$ using the usual Boolean connectives as well as existential and universal quantification over individual and set variables. We write $\phi(x_1, \dots, x_r, X_1, \dots, X_s)$ to indicate that ϕ is a formula with free variables x_1, \dots, x_r and X_1, \dots, X_s , where free variables are variables not bound by quantifiers.

For a formula $\phi(x_1, \dots, x_r, X_1, \dots, X_s)$, a graph G , vertices v_1, \dots, v_r , and sets V_1, \dots, V_r , we write $G \models \phi(v_1, \dots, v_r, V_1, \dots, V_r)$ if ϕ is satisfied in G when \mathcal{E} is interpreted by the adjacency relation $E(G)$, the variables x_i are interpreted by v_i , and variables X_i are interpreted by V_i . We say that a vertex-subset problem Q is *definable in monadic second-order logic* if there exists an MSOL-formula $\phi(X)$ with one free set variable such that $S \subseteq V(G)$ is a feasible solution of problem Q for instance G if and only if $G \models \phi(S)$. For example, an independent set is definable by the formula $\phi_{\text{IS}}(X) = \forall x \forall y (x \in X \wedge y \in X \rightarrow \neg \mathcal{E}(x, y))$.

Theorem 6 (Courcelle [9]). *There is an algorithm that given a MSOL-formula $\phi(x_1, \dots, x_r, X_1, \dots, X_s)$, a graph G , vertices $v_1, \dots, v_r \in V(G)$, and sets $V_1, \dots, V_s \subseteq V(G)$ decides whether $G \models \phi(v_1, \dots, v_r, V_1, \dots, V_s)$ in $\mathcal{O}(f(\text{tw}(G), |\phi|) \cdot n)$ time, for some computable function f .*

Theorem 7. *If a vertex-subset problem Q is definable in monadic second-order logic by a formula $\phi(X)$, then Q -MIN-R and Q -MAX-R parameterized by $\ell + \text{tw}(G) + |\phi|$ are fixed-parameter tractable.*

Proof. We provide a proof for Q -MIN-R as the proof for Q -MAX-R is analogous. Given an instance (G, S_s, S_t, k, ℓ) of Q -MIN-R, we build an MSOL-formula $\omega(X_0, X_\ell)$ such that $G \models \omega(S_s, S_t)$ if and only if the corresponding instance is a yes-instance. Since the size of ω will be bounded by a function of $\ell + |\phi|$, the statement will follow from Theorem 6.

As MSOL does not allow cardinality constraints, we overcome this limitation using the following technique. We let $L \subseteq \{-1, +1\}^\ell$ be the set of all sequences of length ℓ over $\{-1, +1\}$ which do not violate the maximum allowed capacity. In other words, given S_s and k , a sequence σ is in L if and only if for all $\ell' \leq \ell$ it satisfies $|S_s| + \sum_{i=1}^{\ell'} \sigma[i] \leq k$, where $\sigma[i]$ is the i^{th} element in sequence σ . We let $\omega = \bigvee_{\sigma \in L} \omega_\sigma$ and

$$\omega_\sigma(X_0, X_\ell) = \exists_{X_1, \dots, X_{\ell-1}} \bigwedge_{0 \leq i \leq \ell} \phi(X_i) \wedge \bigwedge_{1 \leq i \leq \ell} \psi_{\sigma[i]}(X_{i-1}, X_i)$$

where $\psi_{-1}(X_{i-1}, X_i)$ means X_i is obtained from X_{i-1} by removing one element and $\psi_{+1}(X_{i-1}, X_i)$ means it is obtained by adding one element. Formally, we have:

$$\psi_{-1}(X_{i-1}, X_i) = \exists x \, x \in X_{i-1} \wedge x \notin X_i \wedge \forall y \, (y \in X_i \leftrightarrow (y \in X_{i-1} \wedge y \neq x))$$

$$\psi_{+1}(X_{i-1}, X_i) = \exists x \, x \notin X_{i-1} \wedge x \in X_i \wedge \forall y \, (y \in X_i \leftrightarrow (y \in X_{i-1} \vee y = x))$$

It is easy to see that $G \models \omega_\sigma(S_s, S_t)$ if and only if there is a reconfiguration sequence from S_s to S_t (corresponding to X_0, X_1, \dots, X_ℓ) such that the i^{th} step removes a vertex if $\sigma[i] = -1$ and adds a vertex if $\sigma[i] = +1$. Since $|L| \leq 2^\ell$, the size of the MSOL-formula ω is bounded by an (exponential) function of $\ell + |\phi|$. \square

5 Dynamic programming algorithms

Throughout this section we will consider one fixed instance (G, S_s, S_t, k, ℓ) of Π -MIN-R and a nice tree decomposition $\mathcal{T} = (T, \chi)$ of G . Moreover, similarly to the previous section, we will ask, for a fixed sequence $\sigma \in \{-1, +1\}^\ell$, whether $G \models \omega_\sigma(S_s, S_t)$ holds. That is, we ask whether there is a reconfiguration sequence which at the i^{th} step removes a vertex when $\sigma[i] = -1$ and adds a vertex when $\sigma[i] = +1$. The final algorithm then asks such a question for every sequence σ which does not violate the maximum allowed capacity: $|S_s| + \sum_{i=1}^{\ell'} \sigma[i] \leq k$ for all $\ell' \leq \ell$. This will add a factor of at most 2^ℓ to the running time.

5.1 Signatures as equivalence classes

A reconfiguration sequence can be described as a sequence of steps, each step specifying which vertex is being removed or added. To obtain a more succinct representation, we observe that in order to propagate information up from the leaves to the root of a nice tree decomposition, we can ignore vertices outside of the currently considered bag (X_i) and only indicate whether a step has been used by a vertex in any previously processed bags, i.e. a vertex in $V_i \setminus X_i$.

Definition 8. A signature τ over a set $X \subseteq V(G)$ is a sequence of steps $\tau[1], \dots, \tau[\ell] \in X \cup \{\text{used}, \text{unused}\}$. Steps from X are called vertex steps.

The total number of signatures over a bag X of at most t vertices is $(t+3)^\ell$. Our dynamic programming algorithms start by considering a signature with only **unused** steps in each leaf node, specify when a vertex may be added/removed in introduce nodes by replacing **unused** steps with vertex steps ($\tau[i] = \text{unused}$ becomes $\tau[i] = v$ for the introduced vertex v), merge signatures in join nodes, and replace vertex steps with **used** steps in forget nodes.

For a set $S \subseteq V(G)$ and a bag X , we let $\tau(i, S) \subseteq S \cup X$ denote the set of vertices obtained after executing the first i steps of τ : the i^{th} step adds $\tau[i]$ if $\tau[i] \in X$ and $\sigma[i] = +1$, removes it if $\tau[i] \in X$ and $\sigma[i] = -1$, and does nothing if $\tau[i] \in \{\text{used}, \text{unused}\}$.

A valid signature must ensure that no step deletes a vertex that is absent or adds a vertex that is already present, and that the set of vertices obtained after applying reconfiguration steps to $S_s \cap X$ is the set $S_t \cap X$. Additionally, because Π is hereditary, we can check whether this property is at least locally satisfied (in $G[X]$) after each step of the sequence. More formally, we have the following definition.

Definition 9. A signature τ over X is valid if

- (1) $\tau[i] \in \tau(i-1, S_s \cap X)$ whenever $\tau[i] \in X$ and $\sigma[i] = -1$,
- (2) $\tau[i] \notin \tau(i-1, S_s \cap X)$ whenever $\tau[i] \in X$ and $\sigma[i] = +1$,
- (3) $\tau(\ell, S_s \cap X) = S_t \cap X$, and
- (4) $G[X \setminus \tau(i, S_s \cap X)] \in \Pi$ for all $i \leq \ell$.

It is not hard to see that a signature τ over X is valid if and only if $\tau(0, S_s \cap X), \dots, \tau(\ell, S_s \cap X)$ is a well-defined path between $S_s \cap X$ and $S_t \cap X$ in $R_{\min}(G[X], n)$. We will consider only valid signatures. The dynamic programming algorithms will enumerate exactly the signatures that can be extended to valid signatures over V_i in the following sense:

Definition 10. A signature π over V_i extends a signature π over X_i if it is obtained by replacing some **used** steps with vertex steps from $V_i \setminus X_i$.

However, for many problems, the fact that S is a solution for $G[X]$ for each bag X does not imply that S is a solution for G , and checking this ‘local’ notion of validity will not be enough – the algorithm will have to maintain additional information. One such example is the OCT-R problem, which we discuss in Section 5.4.

5.2 An algorithm for VC-R

To process nodes of the tree decomposition, we now define ways of generating signatures from other signatures. The introduce operation determines all ways that an introduced vertex can be represented in a signature, replacing **unused** steps in the signature of its child.

Definition 11. Given a signature τ over X and a vertex $v \notin X$, the introduce operation, $\text{introduce}(\tau, v)$ returns the following set of signatures over $X \cup \{v\}$:

for every subset I of indices i for which $\tau[i] = \text{unused}$, consider a copy τ' of τ where for all $i \in I$ we set $\tau'[i] = v$, check if it is valid, and if so, add it to the set.

In particular $\tau \in \text{introduce}(\tau, v)$ and $|\text{introduce}(\tau, v)| \leq 2^\ell$. All signatures obtained through the introduce operation are valid, because of the explicit check.

Definition 12. Given a signature τ over X and a vertex $v \in X$, the forget operation, returns a new signature $\tau' = \text{forget}(\tau, v)$ over $X \setminus \{v\}$ such that for all $i \leq \ell$, we have $\tau'[i] = \text{used}$ if $\tau[i] = v$ and $\tau'[i] = \tau[i]$ otherwise.

Since $\tau'(i, S_s \cap X \setminus \{v\}) = \tau(i, S_s \cap X) \setminus \{v\}$, it is easy to check that the forget operation preserves validity.

Definition 13. Given two signatures τ_1 and τ_2 over $X \subseteq V(G)$, we say τ_1 and τ_2 are compatible if for all $i \leq \ell$:

- (1) $\tau_1[i] = \tau_2[i] = \text{unused}$,
- (2) $\tau_1[i] = \tau_2[i] = v$ for some $v \in X$, or
- (3) either $\tau_1[i]$ or $\tau_2[i]$ is equal to **used** and the other is equal to **unused**.

For two compatible signatures τ_1 and τ_2 , the join operation returns a new signature $\tau' = \text{join}(\tau_1, \tau_2)$ over X such that for all $i \leq \ell$ we have, respectively:

- (1) $\tau'[i] = \text{unused}$,
- (2) $\tau'[i] = v$, and
- (3) $\tau'[i] = \text{used}$.

Since $\tau' = \text{join}(\tau_1, \tau_2)$ is a signature over the same set as τ_1 and differs from τ_1 only by replacing some **unused** steps with **used** steps, the join operation preserves validity, that is, if two compatible signatures τ_1 and τ_2 are valid then so is $\tau' = \text{join}(\tau_1, \tau_2)$.

Let us now describe the algorithm. For each $i \in V(T)$ we assign an initially empty table A_i . All tables corresponding to internal nodes of T will be updated by simple applications of the introduce, forget, and join operations.

Leaf nodes. Let i be a leaf node, that is $X_i = \{v\}$ for some vertex v . We let $A_i = \text{introduce}(\tau, v)$, where τ is the signature with only **unused** steps.

Introduce nodes. Let j be the child of an introduce node i , that is $X_i = X_j \cup \{v\}$ for some $v \notin X_j$. We let $A_i = \bigcup_{\tau \in A_j} \text{introduce}(\tau, v)$.

Forget nodes. Let j be the child of a forget node i , that is $X_i = X_j \setminus \{v\}$ for some $v \in X_j$. We let $A_i = \{\text{forget}(\tau, v) \mid \tau \in A_j\}$.

Join nodes. Let j and h be the children of a join node i , that is $X_i = X_j = X_h$. We let $A_i = \{\text{join}(\tau_j, \tau_h) \mid \tau_j \in A_j, \tau_h \in A_h, \text{ and } \tau_j \text{ is compatible with } \tau_h\}$.

The operations were defined so that the following lemma holds by induction. The theorem then follows by making the algorithm accept when A_{root} contains a signature τ such that no step of τ is **unused**.

Lemma 14 (*). For $i \in V(T)$ and a signature τ over X_i , $\tau \in A_i$ if and only if τ can be extended to a signature over V_i that is valid.

Theorem 15 (*). VC-R and IS-R can be solved in $\mathcal{O}^*(4^\ell(t+3)^\ell)$ time on graphs of treewidth t .

5.3 VC-R in planar graphs

Using an adaptation of Baker’s approach for decomposing planar graphs [1], also known as the *shifting technique* [4, 11, 14], we show a similar result for VC-R and IS-R on planar graphs. The idea is that at most ℓ elements of a solution will be changed, and thus if we divide the graph into $\ell + 1$ parts, one of these parts will be unchanged throughout the reconfiguration sequence. The shifting technique allows the definition of the $\ell + 1$ parts so that removing one (and replacing it with simple gadgets to preserve all needed information) yields a graph of treewidth at most $3\ell - 1$.

Theorem 16 (*). *VC-R and IS-R are fixed-parameter tractable on planar graphs when parameterized by ℓ . Moreover, there exists an algorithm which solves both problems in $\mathcal{O}^*(4^\ell(3\ell + 2)^\ell)$ time.*

We note that, by a simple application of the result of Demaine et al. [10], Theorem 16 generalizes to H -minor-free graphs and only the constants of the overall running time of the algorithm are affected.

5.4 An algorithm for OCT-R

In this section we show how known dynamic programming algorithms for problems on graphs of bounded treewidth can be adapted to reconfiguration. The general idea is to maintain a view of the reconfiguration sequence just as we did for VC-R and in addition check if every reconfiguration step gives a solution, which can be accomplished by maintaining (independently for each step) any information that the original algorithm would maintain. We present the details for OCT-R (where Π is the collection of all bipartite graphs) as an example.

In a dynamic programming algorithm for VC on graphs of bounded treewidth, it is enough to maintain information about what the solution’s intersection with the bag can be. This is not the case for OCT. One algorithm for OCT works in time $\mathcal{O}^*(3^t)$ by additionally maintaining a bipartition of the bag (with the solution deleted) [15, 16]. That is, at every bag X_i , we would maintain a list of assignments $X \rightarrow \{\text{used}, \text{left}, \text{right}\}$ with the property that there exists a subset S of V_i and a bipartition L, R of $G[V_i \setminus S]$ such that $X_i \cap S, X_i \cap L$, and $X_i \cap R$ are the **used**, **left**, and **right** vertices, respectively. A signature for OCT-R will hence additionally store a bipartition for each step (except for the first and last sets S_s and S_t , as we already assume them to be solutions).

Definition 17. *An OCT-signature τ over a set $X \subseteq V(G)$ is a sequence of steps $\tau[1], \dots, \tau[\ell] \in X \cup \{\text{used}, \text{unused}\}$ together with an entry $\tau[i, v] \in \{\text{left}, \text{right}\}$ for every $1 \leq i \leq \ell - 1$ and $v \in X \setminus \tau(i, S_s \cap X)$.*

There are at most $(t + 3)^\ell 2^{t(\ell-1)}$ different OCT-signatures. In the definition of validity, we replace the last condition with the following, stronger one:

- (4) For all $1 \leq i \leq \ell - 1$, the sets $\{v \mid \tau[i, v] = \text{left}\}$ and $\{v \mid \tau[i, v] = \text{right}\}$ give a bipartition of $G[X \setminus \tau(p, S_s \cap X)]$.

In the definition of the join operation, we additionally require two signatures to have equal $\tau[i, v]$ entries (whenever defined) to be considered compatible; the operation copies them to the new signature. In the definition of the forget operation, we delete any $\tau[i, v]$ entries, where v is the vertex being forgotten. In the introduce operation, we consider (and check the validity of) a different copy for each way of replacing **unused** steps with v steps and each way of assigning $\{\text{left}, \text{right}\}$ values to new $\tau[i, v]$ entries, where v is the vertex being introduced. As before, to each node we assign an initially empty table of OCT-signatures and fill them bottom-up using these operations. Lemma 14, with the new definitions, can then be proved again by induction.

Theorem 18 (*). OCT-R and IBS-R can be solved in $\mathcal{O}^*(2^{\ell}4^{\ell}(t+3)^{\ell})$ time on graphs of treewidth t .

Similarly, using the classical $\mathcal{O}^*(2^{\mathcal{O}(t \log t)})$ algorithm for FVS and IF (which maintains what partition of X_i the connected components of V_i can produce), we can get the following running times for reconfiguration variants of these problems.

Theorem 19. FVS-R and IF-R can be solved in $\mathcal{O}^*(t^{\ell}4^{\ell}(t+3)^{\ell})$ time on graphs of treewidth t .

6 Conclusion

We have seen in Section 5.4 that, with only minor modifications, known dynamic programming algorithms for problems on graphs of bounded treewidth can be adapted to reconfiguration. It is therefore natural to ask whether the obtained running times can be improved via more sophisticated algorithms which exploit properties of the underlying problem or whether these running times are optimal under some complexity assumptions. Moreover, it would be interesting to investigate whether the techniques presented for planar graphs can be extended to other problems or more general classes of sparse graphs. In particular, the parameterized complexity of “non-local” reconfiguration problems such as FVS-R and OCT-R remains open even for planar graphs.

References

1. B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, Jan. 1994.
2. G. Bauer and F. Otto. Finite complete rewriting systems and the complexity of the word problem. *Acta Informatica*, 21(5):521–540, Dec. 1984.
3. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1–2):1–45, Dec. 1998.
4. H. L. Bodlaender and A. M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, May 2008.
5. P. Bonsma. The complexity of rerouting shortest paths. In *Proc. of Mathematical Foundations of Computer Science*, pages 222–233, 2012.

6. P. Bonsma. Rerouting shortest paths in planar graphs. In *FSTTCS 2012*, volume 18 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 337–349, 2012.
7. L. Cereceda, J. van den Heuvel, and M. Johnson. Connectedness of the graph of vertex-colourings. *Discrete Mathematics*, 308(56):913–919, 2008.
8. L. Cereceda, J. van den Heuvel, and M. Johnson. Finding paths between 3-colorings. *Journal of Graph Theory*, 67(1):69–82, 2011.
9. B. Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Information and Computation*, 85(1):12 – 75, 1990.
10. E. Demaine, M. Hajiaghayi, and K. Kawarabayashi. Algorithmic graph minor theory: secomposition, approximation, and coloring. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 637–646, Oct 2005.
11. E. D. Demaine and M. Hajiaghayi. The bidimensionality theory and its algorithmic applications. *Comput. J.*, 51(3):292–302, 2008.
12. R. Diestel. *Graph Theory*. Springer-Verlag, Electronic Edition, 2005.
13. R. G. Downey and M. R. Fellows. *Parameterized complexity*. Springer-Verlag, 1997.
14. D. Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3):275–291, 2000.
15. S. Fiorini, N. Hardy, B. Reed, and A. Vetta. Planar graph bipartization in linear time. *Discrete Appl. Math.*, 156(7):1175–1180, Apr. 2008.
16. J. Flum and M. Grohe. *Parameterized complexity theory*. Springer-Verlag, 2006.
17. P. Gopalan, P. G. Kolaitis, E. N. Maneva, and C. H. Papadimitriou. The connectivity of boolean satisfiability: computational and structural dichotomies. *SIAM J. Comput.*, 38(6):2330–2355, 2009.
18. M. Grohe. Logic, graphs, and algorithms. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(091), 2007.
19. T. Ito, E. D. Demaine, N. J. A. Harvey, C. H. Papadimitriou, M. Sideri, R. Uehara, and Y. Uno. On the complexity of reconfiguration problems. *Theor. Comput. Sci.*, 412(12-14):1054–1065, 2011.
20. T. Ito, M. Kamiński, and E. D. Demaine. Reconfiguration of list edge-colorings in a graph. *Discrete Applied Mathematics*, 160(15):2199–2207, 2012.
21. M. Kamiński, P. Medvedev, and M. Milanič. Shortest paths between shortest paths. *Theor. Comput. Sci.*, 412(39):5205–5210, 2011.
22. T. Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.
23. A. E. Mouawad, N. Nishimura, V. Raman, N. Simjour, and A. Suzuki. On the parameterized complexity of reconfiguration problems. In *Proc. of the 8th International Symposium on Parameterized and Exact Computation*, pages 281–294, 2013.
24. R. Niedermeier. *Invitation to fixed-parameter algorithms*. Oxford University Press, 2006.
25. E. L. Post. Recursive unsolvability of a problem of Thue. *The Journal of Symbolic Logic*, 12(1):1–11, 1947.
26. W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of computer and system sciences*, 4(2):177–192, 1970.
27. M. Wrochna. Reconfiguration in bounded bandwidth and treedepth, 2014. arXiv:1405.0847.

Appendix

A Details omitted from Section 3

Proof of Lemma 3

Proof. We note that Bauer and Otto's explicit proof for c -balanced Thue systems [2] can easily be adapted to give a 2-balanced Thue system. We include a self-contained proof here for completeness.

Since only words of the same length can be reached by application of rules in a balanced Thue system, it suffices to nondeterministically search all words of the same length to solve the problem in nondeterministic polynomial space. By Savitch's Theorem [26], this places the problem in PSPACE.

Let $M = (\Sigma, Q, q_0, q_{acc}, q_{rej}, \delta)$ be a deterministic Turing Machine working in space bounded by a polynomial $p(|x|)$, where p is a polynomial function and $x \in \Sigma^*$, which accepts any PSPACE-complete language. (By starting from a fixed PSPACE-complete problem we show the word problem to be hard for a certain fixed Thue system; starting from any language in PSPACE we would only show that the more general word problem, where the system is given as input, is PSPACE-complete). Σ is the tape alphabet of M , Q is the set of states, q_0, q_{acc}, q_{rej} are the initial, accepting, and rejecting state respectively, and $\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{\cdot, L, R\}$ is the transition function of M . Let $\$, \epsilon \in \Sigma$ denote the left and right end-markers. We assume without loss of generality that the machine clears the tape and moves its head to the left end when reaching the accepting state.

For any input $x \in \Sigma^*$ we encode a configuration of the Turing Machine by a word of length exactly $p(|x|)$ over the alphabet $\Gamma = \Sigma \cup (\Sigma \times Q) \cup \{\sqcup\}$. If the tape content is $\$a_1a_2 \dots a_n\epsilon$ for some $a_1, \dots, a_n \in \Sigma$, the head's position is $i \in \{0, 1, \dots, n+1\}$ and the machine's state is q , then we define the corresponding word to be the tape content padded with \sqcup symbols and with a_i replaced by (q, a_i) , that is $\$a_1 \dots a_{i-1}(q, a_i)a_{i+1} \dots a_n\epsilon \sqcup \dots \sqcup \in \Gamma^{p(|x|)}$. The initial configuration is then encoded as $s_x = (q_0, \$)x\epsilon \sqcup \dots \sqcup \in \Gamma^{p(|x|)}$ and the only possible accepting configuration is encoded as $t_x = (q_{acc}, \$)\epsilon \sqcup \dots \sqcup \in \Gamma^{p(|x|)}$. Since M never uses more than $p(|x|)$ space on input x , our encoding is well defined for all configurations appearing in the execution of M on x . So M accepts input x if and only if from s_x one reaches the configuration t_x by repeatedly applying the transition function. Such an application corresponds exactly to the following (ordered) string rewriting rules, in the encodings:

- $((q, a)c, (p, b)c)$ for $q \in Q, a, c \in \Sigma$ and $\delta(q, a) = (p, b, \cdot)$,
- $((q, a)c, b(p, c))$ for $q \in Q, a, c \in \Sigma$ and $\delta(q, a) = (p, b, R)$,
- $(c(q, a), (p, c)b)$ for $q \in Q, a, c \in \Sigma$ and $\delta(q, a) = (p, b, L)$.

The transition relation is not symmetric, but since the machine M is deterministic, the configuration digraph (with machine configurations as vertices and the transition function as the adjacency relation) has out-degree one. The configuration t_x (which is a configuration in the accepting state) has a loop, i.e.

a directed edge from t_x to t_x . Therefore from any configuration, t_x is reachable by a directed path if and only if it is reachable by any path. This means that M accepts input x if and only if applying the transition rules to s_x leads to t_x if and only if $s_x \leftrightarrow_R^* t_x$, where R is the symmetric closure of the above rules, i.e., the 2-balanced Thue system over Γ with rules:

- $\{(q, a)c, (p, b)c\}$ for $q \in Q, a, c \in \Sigma$ and $\delta(q, a) = (p, b, \cdot)$,
- $\{(q, a)c, b(p, c)\}$ for $q \in Q, a, c \in \Sigma$ and $\delta(q, a) = (p, b, R)$,
- $\{c(q, a), (p, c)b\}$ for $q \in Q, a, c \in \Sigma$ and $\delta(q, a) = (p, b, L)$.

Since the map $x \mapsto (s_x, t_x)$ is computable in logarithmic space, this proves the word problem of (Γ, R) to be PSPACE-hard. \square

Proof of Lemma 4

Proof. We first need to slightly strengthen Lemma 3 to give a Thue system where only one symbol at a time can be changed. To that aim, it suffices to replace a rule changing two symbols with a sequence of rules using two new intermediary symbols.

Claim 1 *There is a 2-balanced Thue system (Γ, R) whose word problem is PSPACE-complete and such that for every rule $\{a_1a_2, b_1b_2\} \in R$ either $a_1 = b_1$ or $a_2 = b_2$.*

Proof. Let (Σ, R) be the 2-balanced Thue system from Lemma 3. Suppose $\{a_1a_2, b_1b_2\}$ is a rule of R in which $a_1 \neq b_1$ and $a_2 \neq b_2$. We construct a 2-balanced Thue system (Γ, S) with one fewer such rule, preserving PSPACE-completeness of the word problem. The claim then follows inductively.

Let $\Gamma = \Sigma \cup \{X, Y\}$, where X and Y are new symbols which will be used to replace a rule changing two symbols with a sequence of rules changing only one symbol. Let $S = R \setminus \{\{a_1a_2, b_1b_2\}\} \cup \{\{a_1a_2, Xa_2\}, \{Xa_2, XY\}, \{XY, b_1Y\}, \{b_1Y, b_1b_2\}\}$. We show that for any $s, t \in \Sigma^*$ it holds that $s \leftrightarrow_R^* t$ if and only if $s \leftrightarrow_S^* t$, which implies that our construction preserves PSPACE-completeness.

Clearly if $s \leftrightarrow_R^* t$ then $s \leftrightarrow_S^* t$, because replacing a_1a_2 with b_1b_2 can be done in S by replacing a_1a_2 with Xa_2 , then XY , then b_1Y and finally b_1b_2 . Suppose now $s \leftrightarrow_S^* t$ for some $s, t \in \Sigma^*$. Then there is a sequence $s = u_0, u_1, u_2, \dots, u_l = t$ of words $u_i \in \Gamma^*$ such that $u_i \leftrightarrow_S u_{i+1}$. Let $\phi : \Gamma^* \rightarrow \Sigma^*$ be defined by replacing all XY substrings of a word with a_1a_2 , then replacing all remaining X symbols with a_1 and all remaining Y symbols with b_2 . It is easy to check that $\phi(u_i) \leftrightarrow_R \phi(u_{i+1})$ or $\phi(u_i) = \phi(u_{i+1})$. Since $\phi(u_0) = \phi(s) = s$ and $\phi(u_l) = \phi(t) = t$, this implies that $s \leftrightarrow_R^* t$. \square

Let (Γ, S) be the 2-balanced Thue system from Lemma 1 (so if $\{a_1a_2, b_1b_2\} \in S$ then $a_1 = b_1$ or $a_2 = b_2$). Let $S = \{S_1, \dots, S_m\}$.

Let $\sqcup, \$, \ell, x_1, \dots, x_m$ be new symbols, let $\Delta_1 = \{\$, \ell, x_1, \dots, x_m\}$, $\Delta_2 = (\Gamma \cup \{\sqcup\}) \times (\Gamma \cup \{\sqcup\})$, and let $\Delta = \Delta_1 \cup \Delta_2$. We will call Δ_1 *special symbols* and Δ_2 *pair symbols*. Let $H = (\Delta, E)$, where we define $E \subseteq \Delta^2$ as the relation containing the following pairs

- $((a, b), (b, c))$ for any $a, b, c \in \Gamma$,
- $(\$, (\sqcup, a))$ for any $a \in \Gamma$,
- $((a, \sqcup), \epsilon)$ for any $a, b, c \in \Gamma$,
- $((\cdot, a_1), x_i)$,
- $((\cdot, b_1), x_i)$,
- $(x_i, (a_2, \cdot))$,
- $(x_i, (b_2, \cdot))$ for any $\cdot \in \Gamma$ and $i \in \{1, \dots, m\}$ such that $S_i = \{a_1 a_2, b_1 b_2\}$.

Let $(s, t) \in \Gamma^* \times \Gamma^*$ be an instance of the word problem for S , without loss of generality $|s| = |t| = n$. Define $\psi : \Gamma^n \rightarrow \Delta^{n+3}$ as

$$\psi(a_1 a_2 \dots a_n) = \$(\sqcup, a_1)(a_1, a_2)(a_2, a_3) \dots (a_{n-1}, a_n)(a_n, \sqcup)\epsilon$$

It is easy to see that if $s \leftrightarrow_S^* t$ then $\psi(s)$ can be transformed into $\psi(t)$, e.g., applying the rule $S_i = \{a_1 a_2, b_1 a_2\}$ corresponds to replacing $(\cdot, a_1)(a_1, a_2)(a_2, \cdot)$ by $(\cdot, a_1)x_i(a_2, \cdot)$, then $(\cdot, b_1)x_i(a_2, \cdot)$, then $(\cdot, b_1)(b_1, a_2)(a_2, \cdot)$. We will show the other direction, that if $\psi(s)$ can be transformed into $\psi(t)$, then $s \leftrightarrow_S^* t$. Since ψ is computable in logarithmic space, this will imply our claim of PSPACE-completeness.

Indeed, suppose that there is a sequence of H -words $\psi(s) = u_0, u_1, \dots, u_l = \psi(t)$ with $u_j \in \Delta^{n+3}$, such that u_j differs from u_{j+1} only at one position. In any H -word $v = v_1 v_2 \dots v_{n+3} \in \Delta^{n+3}$ there cannot be two consecutive special symbols. We can thus define a word $\phi(v)$ of length n over Γ such that its i th symbol, for $i \in \{1, \dots, n\}$, is the second element of v_{i+1} if v_{i+1} is a pair symbol and the first element of v_{i+2} if v_{i+2} is a pair symbol (either case must hold and if both do, the definitions agree by construction of E). In particular $\phi(\psi(v)) = v$ for any $v \in \Gamma^n$. We argue that $\phi(u_{j-1}) \leftrightarrow_S^* \phi(u_j)$ for $j \in \{1, \dots, l\}$.

Notice that the special symbol $\$$ must precede a pair symbol (\sqcup, \cdot) for some $\cdot \in \Gamma$ and any such pair symbol must be preceded by $\$$. Since only one symbol at a time can be changed, it follows inductively that for each $j \in \{0, \dots, l\}$ the first two symbols of u_j must be $\$(\sqcup, \cdot)$ for some $\cdot \in \Gamma$ and $\$$ appears nowhere else. A similar argument applies to the last two symbols, $(\cdot, \sqcup)\epsilon$ for some $\cdot \in \Gamma$.

Since u_{j-1} and u_j differ at only one position, there are non-empty words $v, w \in \Delta^*$ and symbols $a, b \in \Delta$, $a \neq b$ such that $u_{j-1} = vaw$ and $u_j = vbw$. If a or b is a special symbol then both the last symbol of v and the first symbol of w are pair symbols, so $\phi(u_{j-1}) = \phi(u_j)$. Otherwise, let $a = (a_1, a_2), b = (b_1, b_2)$. Assume without loss of generality that $a_1 \neq b_1$ and $a_2 = b_2$ (the case $a_1 = b_1, a_2 \neq b_2$ is analogous and the case $a_1 \neq b_1, a_2 \neq b_2$ can be split by showing that $\phi(u_{j-1}) \leftrightarrow_S^* \phi(u')$ and $\phi(u') \leftrightarrow_S^* \phi(u_j)$ for $u' = v(b_1, a_2)w$, which can easily be checked to be an H -word). If the last symbol of v is a pair symbol (c, d) , then $d = a_1$ and $d = b_1$, contradicting our assumption. If the last symbol of v is $\$$, then $a_1 = b_1 = \sqcup$. Finally if the last symbol of v is x_i for some $i \in \{1, \dots, m\}$, then S_i must equal $\{ca_1, c'b_1\}$ for some $c, c' \in \Gamma$. Since $a_1 \neq b_1$, we have $c = c'$ and the second-to-last symbol of v must be a pair (\cdot, c) for some $\cdot \in \Gamma \cup \{\sqcup\}$. Thus $\phi(v(b_1, a_2)w)$ is obtained from $\phi(v(a_1, a_2)w)$ by replacing the symbol a_1 at position $|v|$, which is preceded by a c , by the symbol b_1 , that is, $\phi(v(b_1, a_2)w) \leftrightarrow_S \phi(v(a_1, a_2)w)$. \square

B Details omitted from Section 5.2

Proof of Lemma 14

Proof. We first prove a few statements about signature validity. Note that all signatures in the algorithm are obtained through join, forget or introduce operations, which preserve validity and thus for each $i \in V(T)$, the table A_i contains only valid signatures over X_i .

Lemma 20. *If a signature τ over X is obtained from a valid signature by replacing all vertex steps not in X by **used** or **unused** steps, then τ is valid as well.*

Proof. Let τ be obtained from a valid signature τ' over X' by replacing all vertex steps in $X' \setminus X$ by **used** or **unused** steps. First note that $\tau(i, S_s \cap X) = \tau'(i, S_s \cap X') \cap X$. The first three conditions of Definition 9 follow immediately. As Π is hereditary, $G[X' \setminus S] \in \Pi$ implies $G[X \setminus (S \cap X)] \in \Pi$, hence the fourth condition also follows. \square

Lemma 21. *Let G be a graph S, X_1, X_2 be subsets of $V(G)$ such that every edge of $G[X_1 \cup X_2]$ is contained in $G[X_1]$ or $G[X_2]$. If $S \cap X_1$ is a vertex cover of $G[X_1]$, $S \cap X_2$ is a vertex cover of $G[X_2]$ then S is a vertex cover of $G[X_1 \cup X_2]$.*

Proof. Let uv be an edge of $G[X_1 \cup X_2]$. Then it is an edge of $G[X_i]$ for some $i \in \{1, 2\}$. Hence one of u, v must be a member of $S \cap X_i$. Thus every edge of $G[X_1 \cup X_2]$ has an endpoint in S . \square

Corollary 22. *Let τ, τ_1, τ_2 be signatures over X, X_1, X_2 respectively, such that $X = X_1 \cup X_2$ and every edge of $G[X]$ is contained in $G[X_1]$ or $G[X_2]$. Assume furthermore that for all $i \leq \ell$:*

$$\begin{aligned} \tau[i] &= \tau_1[i] \text{ whenever } \tau[i] \in X_1 \text{ or } \tau_1[i] \in X_1 \text{ and} \\ \tau[i] &= \tau_2[i] \text{ whenever } \tau[i] \in X_2 \text{ or } \tau_2[i] \in X_2. \end{aligned}$$

If τ_1 and τ_2 are valid, then so is τ .

Proof. The assumption means that τ and τ_1 agree over all changes within X_1 , that is, $\tau_1(i, S_s \cap X_1) = \tau(i, S_s \cap X) \cap X_1$ (and similarly for τ_2). The first two conditions of Definition 9 for τ follow immediately: if $\tau[i] \in X$ then $\tau[i] \in X_1$ or $\tau[i] \in X_2$, so the statement is equivalent to the first two conditions for τ_1 or for τ_2 . To show the third condition for τ , observe that $\tau(i, S_s \cap X) = (\tau(i, S_s \cap X) \cap X_1) \cup (\tau(i, S_s \cap X) \cap X_2) = \tau_1(i, S_s \cap X_1) \cup \tau_2(i, S_s \cap X_2) = (S_t \cap X_1) \cup (S_t \cap X_2) = S_t \cap X$. For the last condition, it suffices to use Lemma 21 for $S = \tau(i, S_s \cap X)$. \square

We now prove Lemma 14: For $i \in V(T)$ and a signature τ over X_i , $\tau \in A_i$ if and only if τ can be extended to a signature over V_i that is valid. We prove the statement by induction over the tree T , that is, we prove the statement to be true at $i \in V(T)$ assuming we have already proved it for all other nodes in the subtree of T rooted at i . Depending on whether i is a leaf, forget, introduce or join node, we have the following cases.

Leaf nodes. Let v be the only vertex of X_i , that is, $V_i = X_i = \{v\}$. Since $V_i = X_i$, a signature τ over X_i can be extended to a signature valid over V_i if and only if τ is valid and has no **used** steps. That is, if and only if τ has only **unused** and v steps and is valid (over X_i), which happens if and only if $\tau \in A_i$.

Forget nodes. Let j be the child of i , thus $X_i = X_j \setminus \{v\}$ for some $v \in X_j$ and $V_i = V_j$.

For one direction, suppose $\tau \in A_i$ over X_i . Then there is a τ_j in A_j over X_j such that $\tau = \text{forget}(\tau_j, v)$. By inductive assumption, τ_j has an extension π valid over $V_j = V_i$. Since τ_j is obtained from τ by replacing some **used** steps with v steps, π is also an extension of τ . Thus τ has an extension valid over V_i .

For the other direction, suppose τ has an extension π valid over V_i . Then π is obtained from τ by replacing some **used** steps with vertex steps from $V_i \setminus X_i$. Since $V_i \setminus X_i = (V_j \setminus X_j) \cup \{v\}$, we can consider the signature τ_j over $X_j \cup \{v\}$ obtained by only using the replacements with v steps. This signature τ_j can be extended to π by using the remaining replacements, so by inductive assumption $\tau_j \in A_j$. Furthermore, $\text{forget}(\tau_j, v) = \tau$. Thus $\tau \in A_i$.

τ	unused	used	used	X_j
τ_j	unused	used	v	X_j
π	unused	$V_i \setminus X_i$	v	X_j

Introduce nodes. Let j be the child of i , thus $X_i = X_j \cup \{v\}$ for some $v \in X_i$ and $V_i = V_j \cup \{v\}$.

For one direction, suppose $\tau \in A_i$ is a signature over X_i . Then there is a $\tau_j \in A_j$ such that τ can be obtained from τ_j by replacing some **unused** steps with v steps. By inductive assumption τ_j has an extension π_j over V_j that is valid. As π_j can be obtained from τ_j by replacing **used** steps with vertex steps from $V_j \setminus X_j$ and τ has **used** steps at the same positions, we can use the same replacements to obtain an extension π over $V_j \cup \{v\}$ of τ . π agrees with π_j over V_j and with τ over X_i , it is thus valid over V_i by Corollary 22. Therefore τ has an extension over V_i that is valid.

For the other direction, suppose τ has an extension π over V_i that is valid. Let π_j be the signature over $V_j = V_i \setminus \{v\}$ obtained by replacing all v steps of π with **unused** steps. By Lemma 20, π_j is valid. Let τ_j be the signature over $X_j = X_i \setminus \{v\}$ obtained by replacing all v steps of τ with **unused** steps. Then π_j is an extension of τ_j , thus $\tau_j \in A_j$ by inductive assumption. Since π is valid, so is τ (Lemma 20), thus $\tau \in \text{introduce}(\tau_j, v)$ and $\tau \in A_i$.

τ	unused	used	v	X_j
τ_j	unused	used	unused	X_j
π_j	unused	$V_j \setminus X_j$	unused	X_j
π	unused	$V_j \setminus X_j$	v	X_j

Join nodes. Let j, h be the children of i , thus $V_i = V_j \cup V_h$ and we will write X for $X_i = X_j = X_h$.

For one direction suppose $\tau \in A_i$ valid over X . Then there are two compatible signatures $\tau_j \in A_j, \tau_h \in A_h$ such that $\tau = \text{join}(\tau_j, \tau_h)$. By inductive assumption,

they have valid extensions, π_j over V_j and π_h over V_h , respectively. Let I_i, I_j, I_h be the sets of indices of **used** steps in τ, τ_j, τ_h , respectively. By Definition 13, I_i is the sum of disjoint sets I_j, I_h . Since π_j is obtained from τ_j by replacing steps at indices I_j with vertex steps from $V_j \setminus X$ and similarly for π_h , we can define a signature π obtained from τ over $X \cup (V_j \setminus X) \cup (V_h \setminus X) = V_i$ by using both sets of replacements. π is an extension of τ . Moreover, π agrees with π_j over V_j and with π_h over V_h , so by Corollary 22, π is valid over $V_j \cup V_h = V_i$. Therefore τ has a extension over V_i that is valid.

For the other direction, suppose τ has an extension π over V_i that is valid. Let I_j be the set of indices of vertex steps from $V_j \setminus X$ in π and define I_h accordingly. Let τ_j, π_j be obtained from τ, π by replacing all steps at indices I_h by **unused** steps. Since $V_j \cap V_h = X$, π_j is an extension of τ_j over V_j . By Lemma 20 π_j is valid, thus by inductive assumption $\tau_j \in A_j$. Define τ_h, π_h accordingly and observe that $\tau_h \in A_h$. It is easy to see that τ has **used** steps exactly at the indices $I_j \cup I_h$ and τ_j, τ_h have **used** steps exactly at the disjoint sets of indices I_j, I_h , respectively. This implies τ_j, τ_h are compatible and $\tau = \text{join}(\tau_j, \tau_h)$, so $\tau \in A_i$.

		I_j	I_h	
τ	unused	used	used	X
τ_j	unused	used	unused	X
τ_h	unused	unused	used	X
π_j	unused	$V_j \setminus X$	unused	X
π_h	unused	unused	$V_h \setminus X$	X
π	unused	$V_j \setminus X$	$V_h \setminus X$	X

□

Proof of Theorem 15

Proof. Recall that we say $G \models \omega_\sigma(S_s, S_t)$ if there is a reconfiguration sequence (of vertex covers of G) of length exactly ℓ from S_s to S_t , such that the i^{th} step is a vertex removal if $\sigma[i] = -1$ and a vertex addition if $\sigma[i] = +1$. The following lemma states the correctness of the acceptance condition of our algorithm.

Lemma 23. $G \models \omega_\sigma(S_s, S_t)$ if and only if A_{root} contains a signature τ over X_{root} such that no step of τ is **unused**.

Proof. From Lemma 14, we know that A_{root} contains a signature τ over X_{root} such that no step of τ is **unused** if and only if there is a signature π over $V_{\text{root}} = V$ that is valid and such that no step of π is **unused**. This means that π contains only vertex steps and by definition of validity, the corresponding sequence $\pi(0, S_s), \dots, \pi(\ell, S_s)$ is a reconfiguration sequence of length exactly ℓ from S_s to S_t such that the i^{th} step is a vertex removal if $\sigma[i] = -1$ and a vertex addition if $\sigma[i] = +1$. □

It remains to prove the bound on the running time of our algorithm. The number of nodes in T is in $\mathcal{O}(n)$. Checking the compatibility and validity of

signatures can be accomplished in time polynomial in ℓ, t, n . For each node $i \in V(T)$ the table A_i contains at most $(t+3)^\ell$ signatures. Updating tables at the leaf nodes requires $\mathcal{O}^*(2^\ell)$ time, since we check the validity of 2^ℓ signatures obtained from one introduce operation. In the worst case, updating the table of an introduce node requires $\mathcal{O}^*(2^\ell(t+3)^\ell)$ time, i.e. applying the introduce operation on each signature in a table of size $(t+3)^\ell$. For forget nodes, the time spent is polynomial in the maximum size of a table, that is $\mathcal{O}^*((t+3)^\ell)$. Finally, updating the table of a join node can be implemented in $\mathcal{O}^*(2^\ell(t+3)^\ell)$ time by checking for each of the $(t+3)^\ell$ possible signatures all possible ways to split **used** steps among the two children. The algorithm needs to be run for every $\sigma \in \{-1, +1\}^\ell$ that doesn't violate the maximum allowed capacity, giving in total the claimed $\mathcal{O}^*(4^\ell(t+3)^\ell)$ time bound.

Given an instance (G, S_s, S_t, k, ℓ) of IS-R, we can solve the corresponding VC-R instance $(G, V(G) \setminus S_s, V(G) \setminus S_t, n-k, \ell)$ in $\mathcal{O}^*(4^\ell(t+3)^\ell)$ time on graphs of treewidth t . Combining this fact with Proposition 2 yields the result for IS-R. \square

C Details omitted from Section 5.3

Proof of Theorem 16

Proof. Given a plane embedding of a planar graph G , the vertices of G are divided into layers $\{L_1, \dots, L_r\}$ as follows: Vertices incident to the exterior face are in layer L_0 . For $i \geq 0$, we let G' be the graph obtained by deleting all vertices in $L_0 \cup \dots \cup L_i$ from G . All the vertices that are incident to the exterior face in G' are in layer L_{i+1} in G . L_r is thus the last non-empty layer. A planar graph that has an embedding where the vertices are in r layers is called *r-outerplanar*. The following result is due to Bodlaender [3].

Lemma 24 (Bodlaender [3]). *The treewidth of an r -outerplanar graph G is at most $3r - 1$. Moreover, a tree decomposition of width at most $3r - 1$ can be constructed in time polynomial in $|V(G)|$.*

From Lemma 24, we have the following corollary.

Corollary 25 ([3, 4]). *For a planar graph G , we let \mathcal{E} be an arbitrary plane embedding of G and $\{L_1, \dots, L_r\}$ be the collection of layers corresponding to \mathcal{E} . Then for any $i, \ell \geq 1$, the treewidth of the subgraph $G[L_{i+1} \cup \dots \cup L_{i+\ell}]$ is at most $3\ell - 1$.*

We now summarize the main ideas behind how we use the shifting technique. Note that every vertex in $S_s \Delta S_t$ must be touched at least once in any reconfiguration sequence α from S_s to S_t . In other words, $S_s \Delta S_t \subseteq V(\alpha)$. Moreover, we know that $|V(\alpha)|$ is at most ℓ , as otherwise the corresponding VC-R instance is a no-instance. For an arbitrary plane embedding of a planar graph G and every fixed $j \in \{0, \dots, \ell\}$, we let G_j be the graph obtained by deleting all vertices in $L_{i(\ell+1)+j}$, for all $i \in \{0, 1, \dots, \lfloor n/(\ell+1) \rfloor\}$. Note that $tw(G_j) \leq 3\ell - 1$.

Proposition 26. *If there exists a reconfiguration sequence α of length exactly ℓ between two vertex covers S_s and S_t of a planar graph G , then for some fixed $j \in \{0, \dots, \ell\}$ we have $V(\alpha) \subseteq V(G_j)$.*

We still need a few gadgets before we can apply Theorem 15 on each graph G_j and guarantee correctness. In particular, we need to handle deleted vertices and “border” vertices correctly, i.e. vertices incident to the exterior face in G_j .

We solve at most $\lfloor n/(\ell + 1) \rfloor + 1$ instances of the VC-R problem as follows:

1. Find an arbitrary plane embedding of G .
2. For every fixed $j \in \{0, \dots, \ell\}$:
3. Let $G_j^* = G_j$.
4. Let D_j^* denote the set of vertices deleted from G to obtain G_j^* .
5. If $\{S_s \Delta S_t\} \cap D_j^* \neq \emptyset$:
6. Ignore this instance (continue from line 2).
7. Partition D_j^* into $A_j^* = D_j^* \cap \{S_s \cap S_t\}$ and $B_j^* = D_j^* \setminus A_j^*$.
8. Let $S_{s,j}^* = S_s \cap V(G_j^*)$ and $S_{t,j}^* = S_t \cap V(G_j^*)$.
9. If $\{v \in S_{s,j}^* \Delta S_{t,j}^* \mid |N_G(v) \cap B_j^*| > 0\} \neq \emptyset$:
10. Ignore this instance (continue from line 2).
11. For every vertex $v \in A_j^*$:
12. Add an $(\ell + 1)$ -star centered at u to G_j^* .
13. Add u to $S_{s,j}^*$ and $S_{t,j}^*$.
14. For every vertex in $\{v \in \{S_{s,j}^* \cap S_{t,j}^*\} \mid |N_G(v) \cap B_j^*| > 0\}$:
15. Add $\ell + 1$ degree-one neighbors to v in G_j^* .
16. Solve instance $(G_j^*, S_{s,j}^*, S_{t,j}^*, k, \ell)$.

On lines 5 and 6, we make sure that no vertices from the symmetric difference of S_s and S_t lie in the deleted layers of G , as otherwise G_j^* can be ignored, by Proposition 26. Hence, we know that D_j^* can only include vertices common to both S_s and S_t (vertices in $S_s \cap S_t$) and we can partition D_j^* into two sets accordingly (line 7). In the remaining steps, we add gadgets to account for the capacity used by vertices in A_j^* and the fact that the neighbors of any vertex in B_j^* must remain untouched. In other words, we assume that there exists a reconfiguration sequence α from $S_{s,j}^*$ to $S_{t,j}^*$ in $R_{\min}(G_j^*, k)$. Then α is a reconfiguration sequence from S_s to S_t in $R_{\min}(G, k)$ only if:

$$(1) |S_{s,j}^*| + \text{capacity}(\alpha) \leq k - |A_j^*|,$$

where $\text{capacity}(\alpha) = \max_{1 \leq \ell' \leq \ell} (\sum_{i=1}^{\ell'} \text{sign}(\alpha, i))$ and $\text{sign}(\alpha, i)$ is -1 when the i^{th} step of α is a deletion, +1 when it is an addition; and

$$(2) \text{ no vertex deletion in } \alpha \text{ leaves an edge uncovered in } G.$$

To guarantee property (1), we add an $(\ell + 1)$ -star to G_j^* for every vertex in A_j^* then add the center of the star into both $S_{s,j}^*$ and $S_{t,j}^*$ (lines 11, 12, and 13). Therefore, for every value of j we have $|S| = |S_{s,j}^*|$, $|T| = |S_{t,j}^*|$, and $|S_{s,j}^*| + \text{capacity}(\alpha) \leq k - |A_j^*|$. For property (2), we add $\ell + 1$ degree-one neighbors to every vertex in $\{v \in \{S_{s,j}^* \cap S_{t,j}^*\} \mid |N_G(v) \cap B_j^*| > 0\}$ (lines 14 and 15). Those vertices, as well as the centers of the stars, will have to remain untouched in α , as otherwise deleting any such vertex would require more than ℓ additions.

Since adding degree-one vertices and $(\ell+1)$ -stars to a graph does not increase its treewidth, we have $tw(G_j^*) \leq 3\ell - 1$ for all j (Corollary 25). Hence, for each graph G_j^* we can now apply Theorem 15 and solve the VC-R instance $(G_j^*, S_{s,j}^*, S_{t,j}^*, k, \ell)$ in $\mathcal{O}^*(4^\ell(3\ell+1)^\ell)$ time. We prove in Lemma 27 that our original instance on planar G is a yes-instance if and only if $(G_j^*, S_{s,j}^*, S_{t,j}^*, k, \ell)$ is a yes-instance for some fixed $j \in \{0, 1, \dots, \lfloor n/(\ell+1) \rfloor\}$.

Lemma 27. *(G, S_s, S_t, k, ℓ) is a yes-instance of VC-R if and only if $(G_j^*, S_{s,j}^*, S_{t,j}^*, k, \ell)$ is a yes-instance for some fixed $j \in \{0, 1, \dots, \lfloor n/(\ell+1) \rfloor\}$.*

Proof. For (G, S_s, S_t, k, ℓ) a yes-instance of VC-R, there exists a reconfiguration sequence α of length exactly ℓ from S_s to S_t . Then by Corollary 26, we know that for some fixed $j \in \{0, 1, \dots, \lfloor n/(\ell+1) \rfloor\}$ we have $V(\alpha) \subseteq V(G_j^*)$ and $V(\alpha) \cap N_G(B_j^*) = \emptyset$, as otherwise $V(\alpha) \cap B_j^* \neq \emptyset$. By our construction of G_j^* , the maximum capacity constraint is never violated. Therefore, α is also a reconfiguration sequence from $S_{s,j}^*$ to $S_{t,j}^*$.

For the converse, suppose that $(G_j^*, S_{s,j}^*, S_{t,j}^*, k, \ell)$ is a yes-instance for some fixed $j \in \{0, 1, \dots, \lfloor n/(\ell+1) \rfloor\}$ and let α denote the corresponding reconfiguration sequence from $S_{s,j}^*$ to $S_{t,j}^*$. Since the maximum capacity constraint cannot be violated, we only need to make sure that (i) no reconfiguration step in α leaves an uncovered edge in G and that (ii) none of the degree-one gadget vertices are in $V(\alpha)$. For (ii), it is not hard to see that any such vertex must be touched an even number of times and we can delete those reconfiguration steps to obtain a shorter reconfiguration sequence. Moreover, any reconfiguration sequence of length $\ell - x$, where x is even, can be transformed into a reconfiguration sequence of length ℓ by a simple application of the last reconfiguration step and its reversal $\frac{x}{2}$ times. For (i), assume that α leaves an uncovered edge in G . By our construction of G_j^* , such an edge must have one endpoint in B_j^* . But since we added $\ell+1$ degree-one neighbors to every vertex in the neighborhood of B_j^* , this is not possible. \square

Theorem 16 then follows by combining Proposition 2, Lemma 24, Lemma 27, Theorem 15, and the fact that $tw(G_j^*) \leq 3\ell - 1$, for all $j \in \{0, 1, \dots, \lfloor n/(\ell+1) \rfloor\}$. \square

D Details omitted from Section 5.4

Proof of Theorem 18

Proof. The proof of correctness proceeds very similarly as for VC-R, we only need to argue that the strengthened last condition for validity (which uses the additional information about bipartitions in an essential way) is now strong enough to carry through the main inductive proof.

Lemma 28. *If an OCT-signature τ over X is obtained from a valid OCT-signature by replacing all vertex steps not in X by used or unused steps, then τ is valid as well.*

Proof. Let τ be obtained from a valid OCT-signature τ' over X' by replacing all vertex steps in $X' \setminus X$ by **used** or **unused** steps. First note that $\tau(i, S_s \cap X) = \tau'(i, S_s \cap X') \cap X$. The first three conditions of Definition 9 follow immediately. Moreover, if $G[X' \setminus S]$ has a bipartition L, R , then $L \cap X, R \cap X$ is a bipartition of $G[X \setminus (S \cap X)]$, hence the fourth condition also follows. \square

Lemma 29. *Let G be a graph S, X_1, X_2 be subsets of $V(G)$ such that every edge of $G[X_1 \cup X_2]$ is contained in $G[X_1]$ or $G[X_2]$. Let L, R be a partition of $X_1 \cup X_2$. If $L \cap X_1, R \cap X_1$ is a bipartition of $G[X_1 \setminus S]$ and $L \cap X_2, R \cap X_2$ is a bipartition of $G[X_2 \setminus S]$, then L, R is a bipartition of $G[(X_1 \cup X_2) \setminus S]$.*

Proof. Let uv be an edge of $G[(X_1 \cup X_2) \setminus S]$. Then it is contained in $G[X_i]$ for some $i \in \{1, 2\}$. It has no endpoint in $S \cap X_i$, hence it is an edge of $G[X_i \setminus S]$. Thus one endpoint is in $L \cap X_i$ and the other in $R \cap X_i$. In particular every edge of $G[(X_1 \cup X_2) \setminus S]$ has one endpoint in L and the other in R . \square

Corollary 30. *Let τ, τ_1, τ_2 be a OCT-signatures over X, X_1, X_2 respectively, such that $X = X_1 \cup X_2$ and every edge of $G[X]$ is contained in $G[X_1]$ or $G[X_2]$. Assume furthermore that for all $i \leq \ell$:*

$$\begin{aligned} \tau[i] &= \tau_1[i] \text{ whenever } \tau[i] \in X_1 \text{ or } \tau_1[i] \in X_1, \\ \tau[i] &= \tau_2[i] \text{ whenever } \tau[i] \in X_2 \text{ or } \tau_2[i] \in X_2, \\ \tau[i, v] &= \tau_1[i, v] \text{ whenever } v \in X_1 \text{ and } \tau[i, v] \text{ is defined and} \\ \tau[i, v] &= \tau_2[i, v] \text{ whenever } v \in X_2 \text{ and } \tau[i, v] \text{ is defined.} \end{aligned}$$

If τ_1 and τ_2 are valid, then so is τ .

Proof. The assumption implies that τ and τ_1 agree over all changes within X_1 , that is, $\tau_1(i, S_s \cap X_1) = \tau(i, S_s \cap X) \cap X_1$ (and similarly for τ_2). The first three conditions of validity for τ follow as for VC-R. For the last condition, it suffices to use Lemma 29 for $S = \tau(i, S_s \cap X), L = \{v \mid \tau[i, v] = \text{left}\}, R = \{v \mid \tau[i, v] = \text{right}\}$. \square

The following lemma is proved by induction exactly as for VC-R, only with Lemma 28 and Corollary 30 used when validity needs to be argued.

Lemma 31. *For $i \in V(T)$ and an OCT-signature τ over X_i , $\tau \in A_i$ if and only if τ can be extended to an OCT-signature over V_i that is valid.*

The accepting condition is unchanged and its correctness follows from Lemma 31 the same way. It only remains to consider the running time. The number of possible OCT-signatures is $(t+3)^\ell 2^{t(\ell-1)}$ (instead of the $(t+3)^\ell$ for VC-R). In the join operation, we required the new $\tau[i, v]$ entries to be equal and thus the running time is again 2^ℓ times the number of possible OCT-signatures. In the forget operation the algorithm only does a polynomial number of calculations for each of the OCT-signatures. In the introduce operation, for each of the OCT-signatures we consider in the worst case 2^ℓ possible subsets of **unused** steps and 2^ℓ possible assignments of **left** or **right** to new $\tau[i, v]$ entries. The total running time is thus $\mathcal{O}^*(4^\ell (t+3)^\ell 2^{t\ell})$.

Combining the same complementing technique we used for VC-R and IS-R with Proposition 2, the result for IBS-R follows. \square